ソフトウェア工学 71－3
（1990．2．8）

シングルチップマイコンソフトへの簡易スケジューラの導入

西村 朋子

株式会社 東芝 ，システムソフトウェア技術研究所

近年の半導体技術の急速な進歩と共に，家電製品等に組み込まれる小型シングルチップ・マイコンの需要は急速に増加している．それとともに，マイコン・ソフトの需要も急増しており，ソフトの再利用などによる生産性の向上が急務となっている．この分野では，生産コストの増加を抑えるため，小型シングルチップマイコンが多く使われており，メモリサイズ，処理速度等の制約が厳しい．そのため，多くのリアルタイムシステムで採用されているようなリアルタイムＯＳを導入できず，ハードウェアに直接依存したプログラムが多い．本稿では，ソフトの再利用性向上のため，リアルタイムＯＳの一機能であるスケジューラを簡易的にソフトの中に実現することを考え，ソフトの基本設計工程でスケジューラを導入して設計手法を検討した．さらに，電話機組み込みマイコンを事例として設計を行い，利点，問題点を検討した．

the Software Development with the Simple Scheduler of Single-chip Micro Computers

Tomoko Nishimura

Systems & Software Engineering Laboratory, TOSHIBA Corporation
Yanagi-cho 70, Saiwai-ku, Kawasaki, 210 JAPAN

The demand for software in single chip micro computers embedded in systems has been increasing and the improvement of productivity by software reusing is expected. The software in this field are made in the severe restriction of the memory size and the processing speed, and cannot be introduced operation systems to. We propose the design guide of "Simple Scheduler", which is a function of the operation system, to make software structure simple and to separate software from hardware for the purpose of improving software reusability. We describe an example of a software development embedded in a telephone and consider advantages and disadvantages.

1．はじめに

近年の半導体技術の急速な進歩と共に，マイクロコンピュータ（以下，マイコン），及びそのソフトウェア（以下，ソフト）の需要が急増している．家電製品の分野でも，エアコン，電子レンジ，ＶＴＲ，ＴＶなどの高級家電製品はもとより，最近では，トースター，扇風機などの低価格家電製品への組み込みマイコンも増えている．

現在，家電製品に組み込まれているマイコンは，４～８ビットの小型シングルチップ・マイコンが主流である．そのため，メモリサイズ，処理速度等に厳しい制約があり，多くのリアルタイムシステムで採用されているようなリアルタイムＯＳ，リアルタイム・モニタが実現できない．この分野では，マイコンのハードウェア割り込み機能や，Ｉ／Ｏを直接利用するようなソフトが一般的であり，マイコンの機種の変更，次期製品へのバージョンアップなどの場合のソフトの汎用性，再利用性が低い．

本稿では，このように制約の厳しい小型シングルチップ・マイコンの分野で，リアルタイムＯＳの一機能であるスケジューラを導入し，ソフトの標準化，再利用性の向上を図ることを試みた．導入のため，スケジューラの機能を著しく削減しているので，これを簡易スケジューラと呼ぶ．この簡易スケジューラの導入は，小型シングルチップ・マイコン・ソフトの設計過程のうち，基本設計工程での有効な設計手法となると思われる．事例として，電話機の組み込みマイコン・ソフトを取り上げ，簡易スケジューラを導入した場合の利点，問題点を検討する．

2．シングルチップ・マイコン・ソフトの特殊性

本章では，シングルチップ・マイコン，及びそのソフトの特殊性について述べ，それらがソフトの再利用性に与える影響を検討する．

家電製品などに組み込まれるような小型シングルチップ・マイコンは，他の計算機（ＣＰＵ）と比べて以下のような特徴を持つ．

(1)大量生産のため，なによりも低価格であることが望ましい．

(2)(1)により，メモリサイズ（ＲＯＭ，ＲＡＭ）が小さいものが多い．

(3)(1)により，実行速度があまり速くないものが多い．

(4)電子回路の機能を一部，チップ内に実現したものが多い．（電子部品で作るより安いため）

(5)製品の仕様が変更されると，(4)により，マイコンの仕様も変わる場合がある．

(6)家電製品のライフサイクルが短いため，マイコンの仕様変更も早い．

これらの特殊性により，この分野のソフトは，以下のような特徴がある．

(1)プログラム・サイズ，データ・サイズをできる限り小さくする必要がある．

(2)処理時間をできるだけ短くするようなプログラムにする必要がある．

(3)(1)(2)により，ＯＳ等の導入が不可能なので，直接，チップのハードウェアに依存したプログラムになる．

(4)マイコンなど周囲のハードウェアの仕様が変わると，ソフトの仕様も変わる．

(5)家電製品のライフサイクルが短いため，ソフトの仕様変更も早い．

マイコンの仕様変更，あるいは製品のバージョンアップによる頻繁なソフトの仕様変更に対応するには，ソフトの再利用が有効である．しかし，ハードウェア（チップのメモリ，割り込み，Ｉ／Ｏ，周辺回路等）に直接依存したプログラムとなっている，あるいはプログラムや処理時間を短くするために構造化プログラミングが行われていない場合が多く，再利用性は低い．

ハードウェア隠蔽には，リアルタイムＯＳ，リアルタイム・モニタが有効である．プログラムの構造化の観点からみても，プログラムをタスクの集合と見なし，多重プログラミング（マルチ・プログラミング）を実現するＯＳを導入することは，有効であると思われる．

しかし，ＯＳの導入はほとんど不可能であると言われているこの分野で，再利用性向上のために敢えて導入するには，リアルタイムＯＳの機能の大半を削除した簡易ＯＳ的なものを考えなければならない．本稿では，ＯＳの最も核となるスケジューラの部分だけを簡易な形で実現することを試みた．次章にその詳細を述べる．

3．簡易スケジューラの提案

本章では，シングルチップ・マイコン・ソフトにおける，タスクのスケジューリングについて，現状の問題点を検討する．その解決策の一つとして，簡易スケジューラの導入を提案し，その場合の設計手法を述べる．

3．1．現状の問題点

前章に述べた特徴により，この分野では，多重プログラミングがうまく行われていないソフトも多い．

例えば，図1は，全ての処理をメイン・ルーチンで処理する場合である．処理A→処理B→処理C→処理A→･･･の順に処理が繰り返されるが，入力x，出力yに関する時間要件（入力xは何ミリ秒に1回チェックしなければいけないなど）を満たすためには，この繰り返しの1回が一定時間で終了しなければならない．このため，各処理の内容を圧縮し処理時間を短くする，繰り返しが早く終わり過ぎないように無駄命令を挿入する，などの工夫が行われる．

しかし，このようなプログラムでは，再利用性は非常に低い．例えば，新たに処理Dが加わる場合，入力x，出力yに関する時間要件が変更される場合は，プログラム全体を書き直さなければならない．

図2は，ハードウェアによるタイマ割り込みを利用した例である．タイマ割り込みは，一定周期Tで発生し，メイン処理は強制的に中断させられ，タイマ割り込み処理が実行される．割り込み処理が終了すると，中断していたメイン処理が再開される．

外部の時間要件により，処理Aは，周期Tで実行しなければならないが，その処理時間はTに比べ充分に短いと仮定する．また，処理Bは，周期3Tおきの実行でよいが，その処理時間は，Tと同程度，またはTより大きいとする．

最も簡単な実現方法は，割り込み処理の中で，まず処理Aを実行し，次に3回に1回だけ処理Bを実行するというものである．しかし，処理Bを実行している間に，次のタイマ割り込みが発生してしまう可能性があり，その場合，処理Aが時間要件を満たせないことになる．

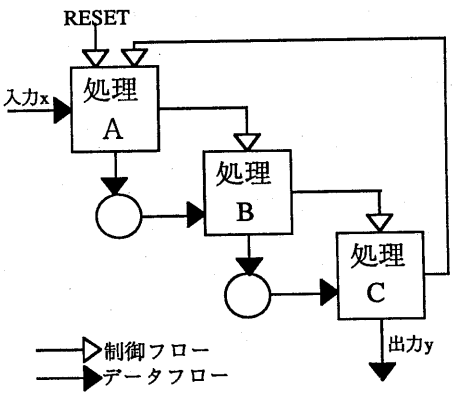


図1　悪い例(1)

その解決策の一つとして，図2では，処理Bを3等分して処理B1，処理B2，処理B3としている．割り込み発生の1回目は処理Aの後処理B1，2回目は処理A→処理B2，3回目は処理A→処理B3，･････と繰り返すことにより，処理Bを完了するようになっている．

このプログラムも，再利用性の点で問題が多い．例えば処理Bの時間要件が変更になった場合，または，仕様の変更による修正で，処理B1の内容が短くなり，処理B2の内容が長くなってしまった場合，処理Bの実現を，全く新しく考え直さなければいけない．

3．2．簡易スケジューラの提案

以上のように，ハードウェアに直接依存したプログラムは，再利用性の面で問題がある．これを解決するためには，まず，

「プログラムをいくつかの機能（＝処理）の単位に分割する」

（例：入力x処理タスク，出力y処理タスクなど）

ということが不可欠である．言い替えれば，処理時間の調整のためだけに分割をしてはいけない．

また，こういう，タスクによる多重プログラミングを実現するために，

「タスク制御を行う部分（スケジューラ）を作り，各タスクを交互に動作させるタイミングなどは，一切制御させる」

ということも必要である．しかし，前章でも述べたように，メモリサイズ，実行速度等の制約から，多機能なスケジューラは，実現できない．そこで，

「スケジューラの機能のうち，個々のプログラムに必要な機能だけを持ったスケジューラを作る」

ことを考える．以下，これを簡易スケジューラと呼ぶ．

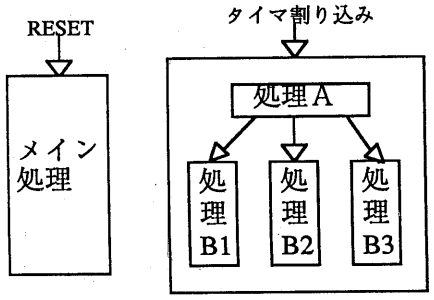


図2　悪い例(2)

簡易スケジューラは，個々のプログラムに応じて，個々に作られるべきであり，言い替えれば，プログラムの一部である．本来のプログラムの部分の再利用性向上のため，簡易スケジューラそのものの再利用性の低さは，無視することとする．

3．3．簡易スケジューラの機能

簡易スケジューラの持つべき機能について検討する．以下の機能をどれだけ盛り込んだものにするかは，個々のプログラムによって違う．

(1)スケジューラの起動

基本的に，タイマ割り込みで周期的に起動し，各タスクの状態を監視するものとする．その他，リセット時や，外部割り込み時なども起動する必要がある場合もある．

(2)タスクの起動条件

各タスクの起動条件は，時間を基本とするのが一般的である．

その他，外部割り込み待ち，イベント待ちで起動することが必要な場合は，その機能を盛り込むこともできる．

(3)タスク数

タスク数が実行中に変化しない場合は，タスク数を固定してもよいが，動的にタスクを生成，消滅させる場合は，タスク数も可変にする必要がある．その場合，ＴＣＢ（タスク・コントロール・ブロック：各タスクを制御するために必要な，スケジューラのデータ群）の大きさも可変にする必要がある．但し，小型のシングルチップ・マイコン・ソフトの分野では，タスク数を可変にする場合は少ないと思われる．

(4)タスクの優先度

タスクの起動に優先順位がある場合，優先度の順に起動するという機能を設ける．また，その優先度を動的に変更する可能性がある場合は，その機能を設ける

(5)タスクの待機

各タスクが，ある程度の処理を完了した後，スケジューラに制御を返すための機能も必要である．このとき，次のタイマ割り込み発生まで他のタスクを動作させないのが，最も単純な方法である．この時間のスキに他のタスクを起動する場合は，その機能も付加しなければならない．

(6)優先実行

優先度の高いタスクが起動されるべき時刻がきたときに，優先度の低いタスクが実行中だった場合，優先度の低いタスクを強制的に中断させて，優先度の高いタスクを起動させるという機能を持たせることもできる．これは，外部の時間要件が厳密である場合に有効である．

(7)Ｉ／Ｏドライバ

複数のタスクで共通に使用するＩ／Ｏルーチンなどは，Ｉ／Ｏドライバとして，スケジューラの中にその機能を持たせることもできる．

なお，共通データの排他制御は，簡易化のため，考慮しないものとする．

3．4．設計手順

簡易スケジューラを導入した場合の基本設計の設計手順を，以下に述べる．

[STEP1]仕様の分析

機能仕様をはじめ，Ｉ／Ｏの時間要件，外部割り込みの使用の有無，周辺機器との通信，ハードウェアの仕様（タイマの性能，メモリの容量など）について，分析する．

とくに，機能仕様と，Ｉ／Ｏの時間要件は，タスク分割の重要なポイントとなるので，注意する．

[STEP 2]タスク分割

おおまかな機能のかたまりによって，ソフト全体を分割する．時間要件の周期が同じもの，外部割り込みなど同じ条件で起動するものを，１タスクにまとめてもよい．タスク数が多すぎると，タスク切り替え処理に時間がかかり，簡易スケジューラのオーバヘッドが大きくなるので，注意しなければいけない．

[STEP 3]簡易スケジューラの機能決定

３．３．に述べたように，どれだけの機能を盛り込むかを検討する．

[STEP 4]簡易スケジューラの設計

決定した機能を実現するように，簡易スケジューラの構造，ＴＣＢを設計する．簡易スケジューラのオーバヘッドを小さくするように，できるだけ処理時間を短くする．

[STEP 5]ソフトウェア構造の決定

設計された簡易スケジューラを元に，各タスク，タ

スク間データを設計し，データの受け渡し規則を決定する．

この基本設計から，詳細設計工程で，各タスク，及び簡易スケジューラをそれぞれ詳細設計する．

３．５．実現性の問題

ＯＳを導入できないシングルチップ･マイコン･ソフトで，簡易スケジューラを本当に導入できるかどうかは，個々の場合によると思われる．また，簡易スケジューラの時間的オーバヘッドを減らすため，タスク数をあまり多くしない，スケジューラの起動周期をあまり小さくしないなどの工夫も必要である．

しかし，簡易スケジューラを導入しないプログラムでは，スケジューラの行うべき仕事が，プログラムのあちこちに分散していたはずであるから，それらを抜き出すことにより，本来のプログラムの部分のメモリサイズ，実行時間は逆に減少することが予測される．

簡易スケジューラを導入できるかどうかの判断は，基本設計の時点で，簡易スケジューラによるメモリサイズの増加，時間的オーバヘッドを試算し，実現の可能性を検証する必要がある．簡易スケジューラの設計が終了すれば，おおまかな試算は可能なはずであるが，現在は，この検証に対する明確な指針はない．

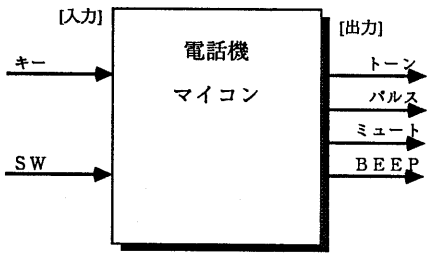


図3　電話機組み込みマイコンシステム図

４．簡易スケジューラの導入例

本章では，電話機組み込みマイコンソフトを例にとり，簡易スケジューラを導入した場合の設計について論じ，その結果を評価する．

４．１．事例の仕様分析

図3に，電話機組み込みマイコンのシステム図を示す．

主な入力は，キー入力とＳＷ入力の２つ，主な出力は，トーン，パルス，ミュート出力などのダイヤル関係の出力と，ＢＥＥＰ音出力の２系統がある．

入力の時間要件は，キー入力，ＳＷ入力ともほぼ同じ周期で，他の出力などとは何の拘束もない．

ダイヤル出力は，出力するデータによって，数個の出力が，数種類のタイミング･パターンで出力され，相互に関連している．時間について，最も厳密さを要求される部分でもある．

ＢＥＥＰ出力は，ダイヤル出力とも，入力とも，直接は関係のないタイミングで出力される．

内部処理としては，短縮ダイヤルの処理，リダイヤルの処理など，かなり複雑で，量の多い処理が必要である．

４．２．タスク分割

以上から，このプログラムは，図4のように，４つのタスクに，タスク分割した．キー入力とＳＷ入力の処理は１タスクとし，２系統の出力をそれぞれタスクとして，内部処理も１タスクとした．

４．３．簡易スケジューラの機能の検討

(1)スケジューラの起動

最も厳密さを要求されているダイヤル出力の，各条件の最大公約数を周期とした．この周期の倍数で，他のタスクも制御できることを確認した．

(2)タスクの起動条件

このプログラムでは，起動条件は，時間とする．ダイヤル出力で数種類のタイミングを要求されるため，起動周期は可変とする．（但し，簡易スケジューラの起動周期の倍数）

(3)タスク数

このプログラムでは，動的にタスク数を変更する必要はないので，４個で固定とする．

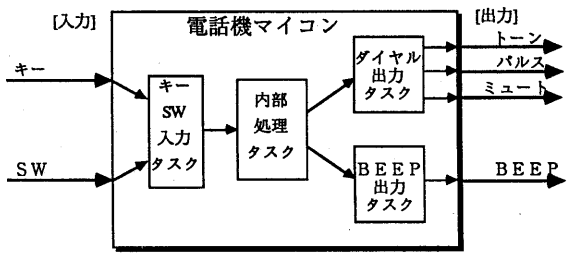


図4　タスク分割後

(4)タスクの優先度

ダイヤル出力タスクを１位，内部処理タスクを最下位とする．キーＳＷ入力タスクとＢＥＥＰ出力タスクでは，起動周期の短いキーＳＷ入力タスクを上位とする．優先度を動的に変更する必要はないので，優先度は固定とする．

(5)タスクの待機

タスクがある程度の処理を終了して待機し，次にスケジューラが動作するまでの間，ＣＰＵを遊ばせておくのはもったいないので，他のタスクを動かすことにする．但し，優先度最下位の内部処理タスクは，他のタスクが動いていないときには常に動くこととし，待機機能は持たせない．

(6)優先実行

ダイヤル出力を正確に実現するため，優先実行を取り入れる．これにより，中断させる場合のレジスタ，スタックなどの各タスクの内部状態の保存機能が必要となる．

(7)Ｉ／Ｏドライバ

入力処理，出力処理は，すべてタスクとしたので，Ｉ／Ｏドライバは用意しない．

(8)その他

スケジューラの内部機能として以下の２つを実現した．

○リセット処理機能

リセット時（電源ON）の初期化処理．

○外部割り込み処理機能

受話器が置かれた時，
上げられた時の処理．

４．４．簡易スケジューラの設計

前節の機能を満たすように，簡易スケジューラの設計を行った．内部構造は，図5のように，

(1)リセット処理部
(2)タイマ割り込み処理部
(3)外部割り込み処理部
(4)タスク待機処理部
(5)起動タスク選択部

の５つの部分から構成される．

このスケジューラによって制御される各タスクの状態を図6に示す．各タスクは，実行，実行可能，待機，の３状態のいずれかを持ち，実行タスクは常に１タスクのみである．

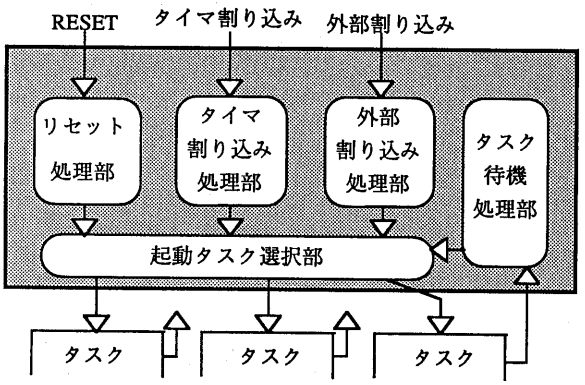


図5　簡易スケジューラの内部構成

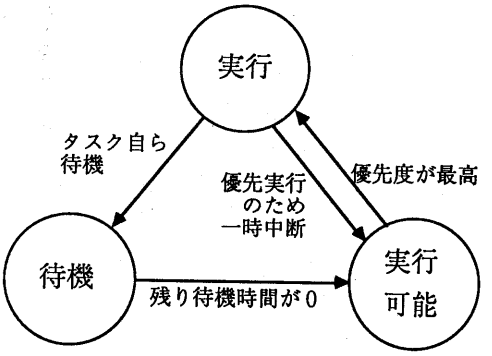


図6　各タスクの制御状態

フ	カウンタ	レジスタ	退　避	エリア	SP退避エリア	
ラ		(A-reg)	(H-reg)	(L-reg)		ダイヤル出力タスク用
グ						キーＳＷ入力タスク用
↑						ＢＥＥＰ出力タスク用
						内部処理タスク用

(実行状態を示す)

表1　簡易スケジューラのＴＣＢ(Task Control Block)

実行タスクは，自ら待機するときは(4)タスク待機処理部を通じて，待機状態に入る．(2)タイマ割り込み処理部は，タイマ割り込みで動作し，実行中のタスクがあれば強制的に実行可能状態にする．その後，待機タスクの残り時間をカウントし，もし0になったら実行可能状態とする．(1)～(4)の後は，(5)起動タスク選択部で，実行可能タスクのうち，最も優先度が高いタスクを起動する．なお，優先度が最下位の内部処理タスクは，待機機能を持たないので，待機状態になることはありえない．

このスケジューラのＴＣＢを表1に示す．

４．５．ソフトウェア構造の決定

前節の簡易スケジューラを導入して基本設計を行った結果の，ソフトウェア構造を図7に示す．

４．６．実現可能性の検討

この時点で，スケジューラのプログラム･サイズ，ＴＣＢのサイズを試算し，既存のメモリ（ＲＯＭ，ＲＡＭ）に納まることを確認した．また，時間的オーバヘッドについても，簡易スケジューラと各タスクのＣＰＵ使用率の最大値を概算し，合計でも100%を超えないことを確認した．

４．７．結果と考察

この基本設計を元にして，電話機組み込みマイコン･ソフトの設計を行った．その結果とその考察について，以下に述べる．

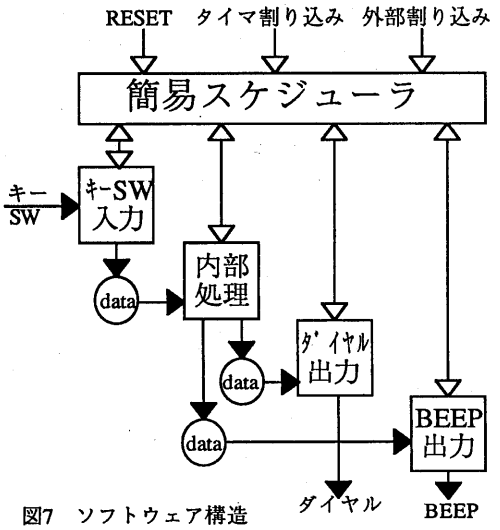


図7　ソフトウェア構造

(1)設計･プログラムの容易さ，再利用性

機能単位でのタスク分割を行ったことにより，設計が容易になったばかりでなく，変更，修正時に影響範囲が小さくて済み，デバッグ作業の工数も減少した．また，次期開発時にも，タスク単位の再利用が容易であろうと思われる．

また，優先実行の機能を取り入れたため，各タスクの実行位置を記憶するという機能を併せ持つことになった．この機能は，タスク内の任意の場所で待機状態に入るということを可能にし，これは，決定的にプログラムの書き方を変えた．

図8は，その一例である．(a)簡易スケジューラ導入前は，状態変数を用いて，その内容により分岐，処理，状態変数の更新を行い，次に起動された（割り込みが発生するなど）時に次の分岐の処理を行う，というものである．(b)簡易スケジューラ導入後は，順番に行う処理を順番に並べ，簡易スケジューラに制御を返すべき部分に「待機する」という命令を挿入している．この大幅な書き方の変革により，より自然な設計が可能となり，プログラムもわかりやすく整理された．

(2)メモリサイズ

この基本設計に従って設計したプログラムのメモリサイズを，スケジューラ導入前に比較し，表2に示す．これによると，メモリサイズは，プログラム（ＲＯＭ），データ（ＲＡＭ）とも，わずかながら増加しているが，簡易スケジューラのプログラム量，ＴＣＢの容量ほどは増加していないので，元のプログラムの中からスケジューラ機能を果たしている部分が抜き出され，整理されているのがわかる．

(3)時間的オーバヘッド

スケジューラによる処理時間のオーバヘッドは，

MAX　23.1%，　MIN　16.5%

であった．これはスケジューラとしてはかなり大きい数字であるが，今回のプログラムでは，この残りの時間で処理を行うことができた．ＣＰＵ使用率は測定しなかったが，スケジューラは，元のプログラムの中からスケジューラ機能を果たしている部分を抜き出しているだけなので，メモリサイズの場合と同様，ＣＰＵ使用率の大幅な増加はないと思われる．

5．結論

本稿では，家電製品などに組み込まれる小型シングルチップ・マイコン・ソフトの分野で，再利用性の向上を目的として，ＯＳ的概念を導入することを試み，一解決策として簡易スケジューラを提案した．

簡易スケジューラは，スケジューラの機能のうち，個々のプログラムに必要な機能だけを採用し，個々のプログラムの中に作成するものなので，汎用性，再利用性は低い．しかし，簡易スケジューラを導入することによって，各タスクは整理され，再利用性は向上する．小型シングルチップ・マイコン・ソフトの設計過程のうち，基本設計工程の一設計手法として，この簡易スケジューラの導入は有効である．

我々は，電話機組み込みマイコン・ソフトを例にとり，この有効性を実証し，最大の問題点であったメモリサイズ，処理速度の点で，実現が可能であることを証明した．

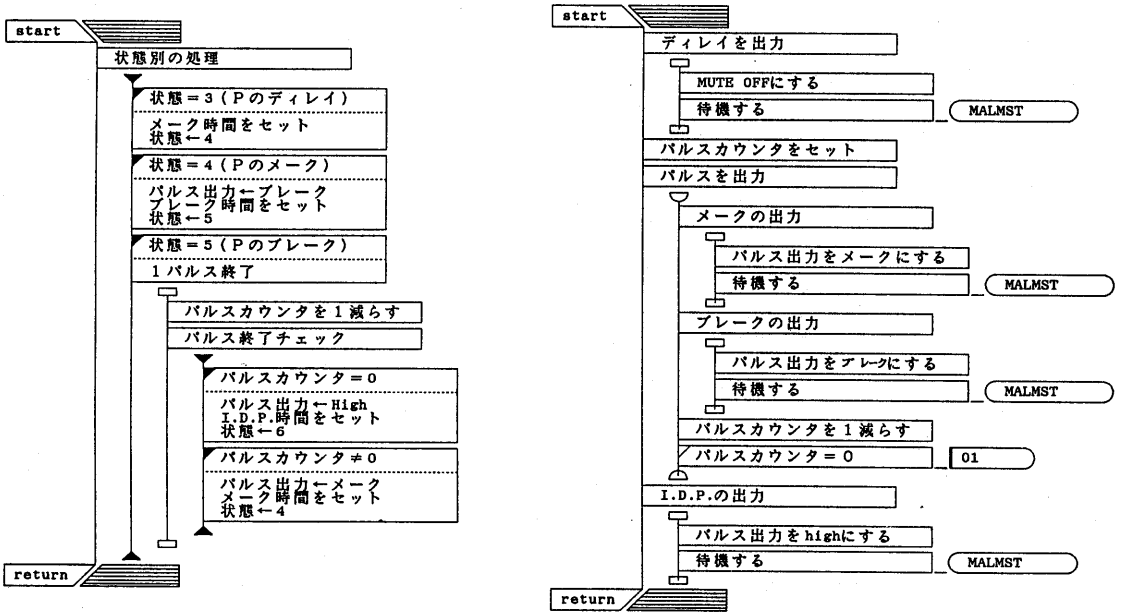
しかし，これはほんの一例にすぎず，全てのシングルチップ・マイコン・ソフトに適用できるわけではなく，基本設計時の，実現可能性の検証が重要である．今後，さらに数多くの事例で，簡易スケジューラを導入して，実現可能性の検証における基準を明確にすることが必要である．

	簡易スケジューラ導入前を100%とする
プログラム・サイズ （簡易スケジューラ）	103.2 % (10.0 %)
データ・サイズ （ＴＣＢ）	106.1 % (12.3 %)

表2　簡易スケジューラ導入によるプログラム量，データ量の変化

［参考文献］

(1)西村他;"シングルチップマイコンソフトの再利用指向設計の一考察－簡易スケジューラの導入－"，情処第37回全国大会，pp.824-825.



(a)簡易スケジューラ導入前　　(b)簡易スケジューラ導入後

図8　同一モジュールの設計の比較